HAL
archives-ouvertes.fr

Realizability of Schedules by Stochastic Time Petri Nets with Blocking Semantics

Loïc Hélouët, Karim Kecir

▶ To cite this version:

Loïc Hélouët, Karim Kecir. Realizability of Schedules by Stochastic Time Petri Nets with Blocking Semantics. 37th International Conference, PETRI NETS 2016, Fabrice Kordon and Daniel Moldt, Jun 2016, Torun, Poland. pp.155-175, 10.1007/978-3-319-39086-4_11 . hal-01379424

HAL Id: hal-01379424
https://hal.inria.fr/hal-01379424

Submitted on 11 Oct 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Realizability of Schedules by Stochastic Time Petri Nets with Blocking Semantics

Loïc Hélouët, Karim Kecir

INRIA Rennes, France
loic.helouet@inria.fr, karim.kecir@inria.fr

Abstract. This paper considers realizability of schedules by stochastic concurrent timed systems. Schedules are high level views of desired executions represented as partial orders decorated with timing constraints, while systems are represented as elementary stochastic time Petri nets. We first consider *logical realizability*: a schedule is realizable by a net $\mathcal{N}$ if it embeds in a time process of $\mathcal{N}$ that satisfies all its constraints. However, with continuous time domains, the probability of a time process that realizes a schedule is null. We hence consider *probabilistic realizability* up to α time units, that holds if the probability that $\mathcal{N}$ logically realizes S with constraints enlarged by α is strictly positive. Upon a sensible restriction guaranteeing time progress, logical and probabilistic realizability of a schedule can be checked on the finite set of symbolic prefixes extracted from a bounded unfolding of the net. We give a construction technique for these prefixes and show that they represent all time processes of a net occurring up to a given maximal date. We then show how to verify existence of an embedding and compute the probability of its realization.

1 Introduction

Correct scheduling of basic operations in automated systems (manufacturing or transport systems,...) is a way to manage at best available resources, avoid undesired configurations, or achieve an objective within a bounded delay. Following a predetermined schedule is also a way to meet QoS objectives. For instance, changes to predetermined schedules in metro networks may cause congestion and reduce QoS. Schedules provide high-level views for correct ordering of important operations in a system, consider time issues and provide optimal dates for a production plan. They can be seen as partial orders among basic tasks, decorated with dates and timing constraints, that abstract low-level implementation details.

Designing a correct and optimal schedule for a system is a complex problem. Occurrence dates of events can be seen as variables, and correct and optimal schedules as optimal solutions (w.r.t. some criteria) for a set of constraints over these variables. Linear programming solutions have been proposed to optimize scheduling in train networks [8, 9]. The size of models for real systems that run for a full day call for approximated solutions usually provided by experts. When a high-level schedule and the low-level system that implements it are designed in a separate way, nothing guarantees that the system is able to realize the expected

schedule. This calls for tools to check realizability of a schedule by a system. One can notice that optimal and realizable schedules are not necessarily robust if they impose tight realization dates to systems that are subject to random variations (delays in productions, faults...). In metro networks, trains delays are expected and are part of the normal behavior of the system. To overcome this problem, metro schedules integrate small recovery margins that avoid the network performance to collapse as soon as a train is late. Note also that for systems where time issues are defined with continuous variables, the probability to execute a given event at a precise date is zero. Furthermore, being able to realize a schedule does not mean that the probability to meet optimal objectives is high enough. Beyond logical realizability, a schedule shall hence be considered as realizable if it can be approached with a significant probability.

This paper addresses realizability of schedules by stochastic timed systems. We define schedules as labeled partial orders decorated with dates and timing constraints, and represent systems with elementary stochastic time Petri nets (STPN for short), a model inspired from [13]. We particularly emphasize on resources: non-availability of a resource (represented by a place) may block transitions. This leads to the definition of a blocking semantics for STPNs that forbids firing a transition if one of its output places is filled. We then propose a notion of realizability: a schedule S is *realizable* by an STPN $\mathcal{N}$ if S embeds in a symbolic process of $\mathcal{N}$ that meets constraints of S. We prove that upon some reasonable time progress assumption, realizability can be checked on a finite set of symbolic processes, obtained from a bounded untimed unfolding [16, 12] of $\mathcal{N}$. Symbolic processes are processes of the unfolding with satisfiable constraints on occurrence dates of events. A symbolic framework to unfold time Petri nets was already proposed in [4, 6] but blocking semantics brings additional constraints on firing dates of transitions. Embedding of a schedule in a process of $\mathcal{N}$ only guarantees *logical realizability*: the probabilty of a time process in which one event is forced to occur at a precise date is 0. We use transient analysis of STPNs [13] to compute the probability that a schedule is realized by a symbolic time process of $\mathcal{N}$ up to an imprecision of δ. This allows to show that $\mathcal{N}$ realizes $S \pm \delta$ with strictly positive probability, and then define a notion of *probabilistic realizability*.

The paper is organized as follows: Section 2 introduces schedules and our variant of stochastic time Petri nets with blocking semantics. Section 3 defines a notion of symbolic processes. Section 4 shows how to verify that a schedule is compatible with at least one process of the system and measure the probability of such realization. Due to lack of space, proofs and several technical details are provided in an extended version available at `hal.inria.fr/hal-01284682`.

2 Schedules and Stochastic Time Petri Nets

A schedule describes causal dependencies among tasks, and timing constraints on their respective starting dates. Schedules are defined as decorated partial orders. We allow timing constraints among tasks that are not causally related.

Definition 1 (schedule). *A* schedule *over a finite alphabet $\mathcal{A}$ is a quadruple $S \triangleq \langle N, \to, \lambda, C\rangle$ where N is a set of nodes, $\to \subseteq N \times N$ is an acyclic precedence relation, $\lambda : N \to \mathcal{A}$ is a labeling of nodes, and $C : N \times N \nrightarrow \mathbb{Q}_{>0}$ is a partial function that associates a time constraint to pairs of nodes. A* dating function *for a schedule S is a function $d : N \to \mathbb{Q}_{\geq 0}$ that satisfies all constraints of C and $\to$: $\langle n, n'\rangle \in \to$ implies $d(n') \geq d(n)$, and $C(n, n') = x$ implies $d(n') - d(n) \geq x$.*

This model for schedules is inspired from [8, 9]. Intuitively, if $C(n, n') = x$, then n' cannot occur earlier than x time units after n, and if $\langle n, n'\rangle \in \to$, then n (causally) precedes n'. Constraints model the minimal times needed to perform tasks and initiate the next ones in production cells, the times needed for trains to move from a station to another, etc. A schedule S is *consistent* if the graph $\langle N, \to \cup \{\langle n, n'\rangle \mid C(n, n') \text{ is defined}\}\rangle$ does not contain cycles. Obviously, consistent schedules admit at least one dating function. A frequent approach is to associate costs to dating functions and to find optimal functions that meet a schedule. A cost example is the earliest completion date. Optimizing this cost amounts to assigning to each node the earliest possible execution date. However, these optimal schedules are not the most probable ones. For the earliest completion date objective, if an event n occurs later than prescribed by d, then all its successors will also be delayed. In real systems running in an uncertain environment (e.g., with human interactions or influenced by weather conditions), tight timings are impossible to achieve. Finding a good schedule is hence a trade-off between maximization of an objective and of the likelihood to stay close to optimal realizations at runtime.

We want to check whether a consistent schedule S with its dating function d can be realized by a system. Systems are described with a variant of Petri nets with time and probabilities, namely stochastic time Petri nets [13]. We will show how to check that (S, d) is realizable by an STPN $\mathcal{N}$, and then how to measure the probability that (S, d) is realized by $\mathcal{N}$. Roughly speaking, an STPN is a time Petri net with distributions on firing times attached to transitions. As for Petri nets, the semantics of our model moves tokens from the preset of a transition to its postset. The time that must elapse between enabling of a transition and its firing is sampled according to the distribution attached to the transition. The major difference with [13] is that we equip our STPNs with a blocking semantics. Due to blockings, stochastic time Petri nets are *safe* (1-bounded). This semantics restriction is justified by the nature of the systems we address: in production chains, places symbolize tools that can process only one item at a time. Similarly, when modeling train networks, an important security requirement is that two trains cannot occupy the same track portion, which can only be implemented with such a blocking semantics. Standard time or stochastic Petri nets do not assume a priori bounds on their markings. A way to force boundedness is to add complementary places to the original Petri net and then study it under the usual semantics [7]. However, this trick does not allow to preserve time and probability issues in STPNs with blockings.

For simplicity, we only consider closed intervals of the form $[a, b]$ with $a < b$ and open intervals of the form $[a, +\infty)$. A *probability density function* (PDF) for

a continuous random variable X is a function $f_X : \mathbb{R} \to [0,1]$ that describes the relative likelihood for X to take a given value. Its integral over the domain of X is equal to 1. A *cumulative distribution function* (CDF) $F_X : \mathbb{R} \to [0,1]$ for X describes the probability for X to take a value less than or equal to a chosen value. We denote by Σ_{pdf} the set of PDFs, Σ_{cdf} the set of CDFs, and we only consider PDFs for variables representing durations, i.e., whose domains are included in $\mathbb{R}_{\geq 0}$. The CDF of X can be computed from its PDF as $F_X(x) = \int_0^x f_X(y)\,dy$. A *marking* is a function that assigns 0 or 1 token to each place $p \in P$.

Definition 2 (stochastic time Petri net). *A stochastic time Petri net (STPN for short) is a tuple $\mathcal{N} = \langle P, T, {}^\bullet(), ()^\bullet, m_0, \mathsf{eft}, \mathsf{lft}, \mathcal{F}, \mathcal{W} \rangle$ where P is a finite set of places; T is a finite set of transitions; ${}^\bullet() : T \to 2^P$ and $()^\bullet : T \to 2^P$ are pre and post conditions depicting from which places transitions consume tokens, and to which places they output produced tokens; $m_0 : P \to \{0,1\}$ is the initial marking of the net; $\mathsf{eft} : T \to \mathbb{Q}_{\geq 0}$ and $\mathsf{lft} : T \to \mathbb{Q}_{\geq 0} \cup \{+\infty\}$ respectively specify the minimum and maximum time-to-fire that can be sampled for each transition; and $\mathcal{F} : T \to \Sigma_{\mathsf{pdf}}$ and $\mathcal{W} : T \to \mathbb{R}_{>0}$ respectively associate a PDF and a strictly positive weight to each transition.*

For a given place or transition $x \in P \cup T$, ${}^\bullet x$ will be called the preset of x, and $x^\bullet$ the postset of x. We denote by f_t the PDF $\mathcal{F}(t)$, and by F_t the associated CDF. To be consistent, we assume that for every $t \in T$, the support of f_t is $[\mathsf{eft}(t), \mathsf{lft}(t)]$. This syntax of STPNs is similar to [13], but we equip them with a blocking semantics, defining sequences of discrete transition firings, and timed moves. We will say that a transition t is *enabled* by a marking m iff $\forall p \in {}^\bullet t, m(p) = 1$. We denote by $\mathsf{enab}(m)$ the set of transitions enabled by a marking m.

For a given marking m and a set of places P', we will denote by $m - P'$ the marking that assigns $m(p)$ tokens to each place $p \in P \setminus P'$, and $m(p) - 1$ tokens to each place $p \in P'$. Similarly, we will denote by $m + P'$ the marking that assigns $m(p)$ tokens to each place $p \in P \setminus P'$, and $m(p) + 1$ tokens to each place $p \in P'$. Firing a transition t is done in two steps and consists in: (1) consuming tokens from ${}^\bullet t$, leading to a *temporary* marking $m_{\text{tmp}} = m - {}^\bullet t$, then (2) producing tokens in $t^\bullet$, leading to a marking $m' = m_{\text{tmp}} + t^\bullet$.

The blocking semantics can be informally described as follows. A variable τ_t is attached to each transition t of the STPN. As soon as the preset of a transition t is marked, τ_t is set to a random value ζ_t (called the *time-to-fire* of t, or TTF for short) sampled from $[\mathsf{eft}(t), \mathsf{lft}(t)]$ according to f_t. We will assume that every CDF F_t is strictly increasing on $[\mathsf{eft}(t), \mathsf{lft}(t)]$, which allows to use *inverse transform sampling* to choose a value (see for instance [17] for details). Intuitively, this TTF represents a duration that *must* elapse before firing t once t is enabled. The value of τ_t then decreases as time elapses but cannot reach negative values. When the TTF of a transition t reaches 0, then if $t^\bullet$ is empty in m_{tmp}, t becomes *urgent* and has to fire unless another transition with TTF 0 and empty postset fires; otherwise (if $t^\bullet$ is not empty in m_{tmp}), t becomes *blocked*: its TTF stops decreasing and keeps value 0, and its firing is delayed until the postset of t becomes empty; in the meantime, t can be disabled by the firing of another

transition. The semantics of STPNs is *urgent*: time can elapse by durations that do not exceed the minimal remaining TTF of enabled transitions that are not blocked. If more than one transition is urgent, then the transition that fires is randomly chosen according to the respective weights of urgent transitions. We formalize the semantics of STPNs in terms of discrete and timed moves between *configurations* that memorize markings and TTFs for enabled transitions.

Definition 3 (configuration of an STPN). *A* configuration of an STPN *is a pair* $\mathcal{C}_N \triangleq \langle m, \tau \rangle$ *where* m *is a marking, and* $\tau : \mathsf{enab}(m) \to \mathbb{R}_{\geq 0}$ *is a function that assigns a positive real TTF* $\tau_i \triangleq \tau(t_i)$ *to each transition* t_i *enabled by* m*. A transition* t *is* enabled *in a configuration* $\langle m, \tau \rangle$ *iff it is enabled by* m*.*

Definition 4 (firable and blocked transitions). *A transition* t *is* firable *in* $\langle m, \tau \rangle$ *iff it is enabled by* m*, all places of its postset are empty in* $m - {}^\bullet t$*, and its TTF is equal to* 0*. We denote by* $\mathsf{fira}(\langle m, \tau \rangle)$ *the set of firable transitions of* $\langle m, \tau \rangle$*. A transition* t *is* blocked *in* $\langle m, \tau \rangle$ *iff it is enabled by* m*, its TTF* $\tau(t)$ *is equal to* 0*, and one of its postset places is marked in* $m - {}^\bullet t$*. We denote by* $\mathsf{blck}(\langle m, \tau \rangle)$ *the set of blocked transitions in* $\langle m, \tau \rangle$*.*

Timed moves: A *timed move* $\langle m, \tau \rangle \xrightarrow{\delta} \langle m, \tau' \rangle$ lets a strictly positive duration δ elapse. To be allowed, δ must be smaller or equal to all TTFs of transitions enabled by m and not yet blocked. The new configuration $\langle m, \tau' \rangle$ decreases TTFs of every enabled and non-blocked transition t by δ time units ($\tau'(t) = \tau(t) - \delta$). Blocked transitions keep a TTF of 0, and m remains unchanged.

Discrete moves: A discrete move $\langle m, \tau \rangle \xrightarrow{t} \langle m', \tau' \rangle$ consists in firing a transition t from a configuration $\langle m, \tau \rangle$ to reach a configuration $\langle m', \tau' \rangle$. Discrete moves change the marking of a configuration, and sample new times to fire for transitions that become enabled after the move. To define the semantics of discrete moves, we first introduce newly enabled transitions.

Definition 5 (newly enabled transitions). *Let* m *be a marking and* t *a transition enabled by* m*. A transition* t' *is* newly enabled *after firing of* t *from* m *iff it is enabled by marking* $m' = (m - {}^\bullet t) + t^\bullet$ *and either it is not enabled by* $m - {}^\bullet t$ *or* $t' = t$*. We denote by* $\mathsf{newl}(m, t) \triangleq \mathsf{enab}(m') \cap (\{t\} \cup (T \setminus \mathsf{enab}(m - {}^\bullet t)))$ *the set of transitions newly enabled by firing of* t *from* m*.*

The transition t fired during a discrete move is chosen among all firable transitions of $\langle m, \tau \rangle$. The new marking reached is $m' = (m - {}^\bullet t) + t^\bullet$, and τ' is obtained by sampling a new TTF for every newly enabled transition and keeping unchanged TTFs of transitions already enabled by m and still enabled by m'.

Complete operational rules for STPN moves can be found in the extended version. We will write $\langle m, \tau \rangle \to \langle m', \tau' \rangle$ iff there exists a timed or discrete move from $\langle m, \tau \rangle$ to $\langle m', \tau' \rangle$, and $\langle m, \tau \rangle \xrightarrow{*} \langle m', \tau' \rangle$ iff there exists a sequence of moves leading from $\langle m, \tau \rangle$ to $\langle m', \tau' \rangle$. An initial configuration for $\mathcal{N}$ is a configuration $\langle m_0, \tau_0 \rangle$ where τ_0 attaches a sampled TTF to each transition enabled by m_0.

Consider the STPN $\mathcal{N}_1$ of Figure 1, and suppose that $\mathcal{N}_1$ is in configuration $\langle m, \tau \rangle$, with $m(p_1) = 1$, $m(p_2) = m(p_3) = 0$, $\tau(t_1) = 5.5$. From this configuration, one can let 5.5 time units elapse, and then fire t_1. After this firing, the STPN reaches marking m' with $m'(p_1) = m'(p_2) = 1$, $m'(p_3) = 0$. New TTFs d_1, d_2

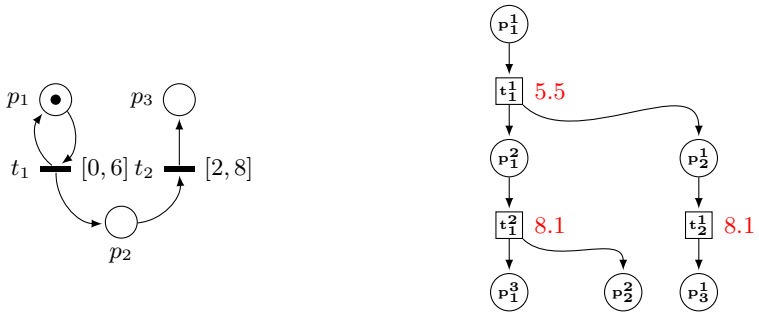

Fig. 1: a) An example STPN $\mathcal{N}_1$ and b) a time process of $\mathcal{N}_1$

are sampled for t_1, t_2, leading to a configuration $\langle m', \tau' \rangle$, where $\tau'(t_1) = d_1$ and $\tau'(t_2) = d_2$. Let us suppose that $d_1 = 1.5$ and $d_2 = 2.6$. Then one can let 1.5 time units elapse, but after this timed move, transition t_1 cannot fire, as place p_2 contains a token. $\mathcal{N}_1$ is hence in a configuration $\langle m', \tau'' \rangle$, where $\tau''(t_1) = 0$, $\tau''(t_2) = 1.1$, and t_1 is blocked. After letting 1.1 time units elapse, transition t_2 can fire, leading to marking $m''(p_1) = m''(p_3) = 1, m''(p_2) = 0$, and t_1 immediately fires at the same date.

Let us now assign probabilities to STPN moves. Randomness in STPNs semantics mainly comes from sampling of TTFs. However, when several transitions are firable from a configuration, weights are used to determine the probability for a transition to fire first. Timed moves are achieved with probability 1: once TTFs are set, there is a unique configuration allowing discrete moves. In a move $\langle m, \tau \rangle \xrightarrow{t} \langle m', \tau' \rangle$, m' is built deterministically, but τ' is obtained by sampling a random value ζ_t for each newly enabled transition t. Each ζ_t is chosen according to CDF F_t, i.e., we have $\mathbb{P}(\zeta_t \leq x) = F_t(x)$ (for any $x \in [\mathsf{eft}(t), \mathsf{lft}(t)]$). When more than one transition is firable from $\langle m, \tau \rangle$, the transition that fires is randomly chosen, and each transition t_k in $\mathsf{fira}(\langle m, \tau \rangle)$ has a probability to fire $\mathbb{P}_{\mathsf{fire}}(t_k) = \mathcal{W}(t_k) / \sum_{t_i \in \mathsf{fira}(\langle m, \tau \rangle)} \mathcal{W}(t_i)$. Note that, as STPNs have continuous probability laws, the probability to choose a particular value ζ_t is the probability of a point in a continuous domain and is hence null. However, in the next sections, we will consider probabilities for events of the form $\tau(t_i) \leq \tau(t_j)$, which may have strictly positive probability.

STPNs define sequences of moves $\rho = (\langle m, \tau \rangle \xrightarrow{e_i} \langle m', \tau' \rangle)_{i \in 1 \dots k}$, where e_i is a transition name in discrete moves and a real value in timed moves. Leaving probabilities for the moment, STPNs can also be seen as generators for timed words over T. A *timed word* over an alphabet $\mathcal{A}$ is a sequence $\langle a_1, d_1 \rangle \dots \langle a_q, d_q \rangle \dots$ in $(\mathcal{A} \times \mathbb{R}_{\geq 0})^*$, where each a_i is a letter from $\mathcal{A}$, each d_i defines the occurrence date of a_i, and $d_1, \dots, d_q$ is an increasing sequence of positive real numbers. Letting $i_1, \dots, i_q$ denote the indices of discrete moves in ρ, we can build a timed word $u_\rho = \langle a_{i_1}, d_1 \rangle \dots \langle a_{i_q}, d_q \rangle \in (T \times \mathbb{R}_{\geq 0})^q$ that associates dates to transitions firings, where $d_1 = \sum_{j < i_1} e_j$, and $d_j = d_{j-1} + \sum_{i_{j-1} < k < i_j} e_k$ for $j \in \{2, \dots, q\}$. The *timed language* of an STPN $\mathcal{N}$ is the set $\mathcal{L}(\mathcal{N})$ of timed words associated with

its sequences of moves. We denote by $\mathcal{L}_{\leq D}(\mathcal{N})$ the set of words in $\mathcal{L}(\mathcal{N})$ whose maximal date is lower than D.

As already highlighted in [2] for TPNs, timed languages give a sequential and interleaved view for executions of inherently concurrent models. A non-interleaved semantics can be defined using *time processes*, i.e., causal nets equipped with dating functions. We recall that *causal nets* are finite acyclic nets of the form $CN \triangleq \langle B, E, {}^\bullet(), ()^\bullet \rangle$, where for every $b \in B$, $|b^\bullet| \leq 1$ and $|{}^\bullet b| \leq 1$. Intuitively, a causal net contains no conflict (pairs of transition with common places in their presets) nor place receiving tokens from more than one transition.

Definition 6 (time process). *A* time process *is a tuple $TP \triangleq \langle CN, \theta \rangle$, where $CN \triangleq \langle B, E, {}^\bullet(), ()^\bullet \rangle$ is a causal net, and $\theta : E \to \mathbb{R}_{\geq 0}$ associates a positive real date to transitions of the net, and is such that $\forall e, e' \in E$ with $e^\bullet \cap {}^\bullet e' \neq \emptyset$ we have $\theta(e) \leq \theta(e')$. In time processes, places in B are called* conditions*, and transitions in E are called* events*. The* depth *of a time process is the maximal number of events along a path of the graph $\langle B \cup E, {}^\bullet() \cup ()^\bullet \rangle$. We will write $e \prec e'$ iff $e^\bullet \cap {}^\bullet e' \neq \varnothing$, and denote by $\preceq$ the transitive and reflexive closure of $\prec$.*

Intuitively, conditions in B represent occurrences of places fillings, and events in E are occurrences of transitions firings. Given an STPN $\mathcal{N}$, for every timed word $u = \langle a_1, d_1 \rangle \dots \langle a_n, d_n \rangle$ in $\mathcal{L}(\mathcal{N})$, we can compute a time process $TP_u = \langle B, E, {}^\bullet(), ()^\bullet, \theta \rangle$. The construction described below is the same as in [2]. It does not consider probabilities and, as the construction starts from an executable word, it does not have to handle blockings either. To differentiate occurrences of transitions firings, an event will be defined as a pair $e \triangleq \langle X, t \rangle$, where t is the transition whose firing is represented e and X is the set of conditions it consumes. Similarly, a condition is defined as a pair $b \triangleq \langle p, e \rangle$, where p is the place whose filling is represented by b, and e is the event whose occurrence created b.

We denote by $\mathsf{tr}(e)$ the transition t attached to an event e, and by $\mathsf{pl}(b)$ the place p associated with a condition b. The flow relations are hence implicit: ${}^\bullet e = \{b \mid e = \langle X, t \rangle \wedge b \in X\}$, and similarly $e^\bullet = \{b \mid b = \langle p, e \rangle\}$, and for $b = \langle p, e \rangle$, ${}^\bullet b = e$ and $b^\bullet = \{e \in E \mid b \in {}^\bullet e\}$. We will then drop flow relations and simply refer to time processes as triples $TP \triangleq \langle B, E, \theta \rangle$. The time process TP_u obtained from a timed word $u = \langle t_1, d_1 \rangle \langle t_2, d_2 \rangle \dots \langle t_k, d_k \rangle \in \mathcal{L}(\mathcal{N})$ is built inductively as follows. We assume a dummy initial event $\perp$ that initializes the initial contents of places according to m_0. We start from the initial process $TP_0 = \langle B_0, E_0, \theta_0 \rangle$ with a set of conditions $B_0 = \{(p, \perp) \mid p \in m_0\}$, a set of events $E_0 = \{\perp\}$, and a function $\theta_0 : \{\perp\} \to \{0\}$.

Let $TP_{u,i} = \langle B_i, E_i, \theta_i \rangle$ be the time process built after i steps for the prefix $\langle t_1, d_1 \rangle \dots \langle t_i, d_i \rangle$ of u, and let $\langle t, d_{i+1} \rangle$ be the $(i+1)^{th}$ entry of u. We denote by $\mathsf{last}(p, E_i, B_i)$ the last occurrence of place p in $TP_{u,i}$, i.e., the only condition $b = \langle p, e \rangle$ with an empty postset. Then, we have $E_{i+1} = E_i \cup \{e\}$, where $e = \langle t, X \rangle$ with $X = \{b \mid b = \mathsf{last}(p, E_i, B_i) \wedge p \in {}^\bullet t\}$ and $B_{i+1} = B_i \cup \{\langle p, e \rangle \mid p \in t^\bullet\}$. We also set $\theta(e) = d_{i+1}$. The construction ends with $TP_u = TP_{u,|u|}$.

Figure 1-b is an example of a time process for STPN $\mathcal{N}_1$. In this example, event t_i^j (resp. condition p_i^j) denotes the j^{th} occurrence of transition t_i (resp. place p_i). This time process corresponds to the time word $u = \langle t_1, 5.5 \rangle \langle t_2, 8.1 \rangle \langle t_1, 8.1 \rangle \in$

$\mathcal{L}(\mathcal{N}_1)$. It contains causal dependencies among transitions (e.g., from t_1^1 to t_2^1). Event t_1^2 cannot occur before t_2^1 as t_1 cannot fire as long as place p_2 is filled. However, this information is not explicit in the process. The timed language $\mathcal{L}(\mathcal{N})$ of a TPN can be reconstructed as the set of linearizations of its time processes. In these linearizations, ordering of events considers both causality and dates of events: e must precede $e' \neq e$ in a linearization of a process if $\theta(e) < \theta(e')$ or if $e \preceq e'$. With blocking semantics, some causality and time-preserving interleavings may not be valid timed words of $\mathcal{L}(\mathcal{N})$: in the process of Figure 1-b, t_1^2 cannot occur before t_2^1, even if both transitions have the same date. A correct ordering among events with identical dates in a process TP_u can however be found by checking that a chosen ordering does not prevent occurrence of other transitions.

3 Unfolding of STPNs

A time process emphasizes concurrency but only gives a partial order view of a *single* timed word. Many time processes of $\mathcal{N}_1$ have the same structure as the process of Figure 1-b, but different dating functions. Indeed, there can be uncountably many time processes with identical structure, but different real dates. It is hence interesting to consider symbolic (time) processes, that define constraints on events dates instead of exact dates. Similarly, to avoid recomputing the structural part of each symbolic process, we will work with *unfoldings*, i.e., structures that contain all symbolic processes of an STPN, but factorize common prefixes. Symbolic unfoldings were introduced for TPNs in [18] and used in [5]. In this section, we show how to unfold STPNs with blockings and extract symbolic processes out of this unfolding. Our aim is to find the minimal structure that represents prefixes of all symbolic processes that embed a schedule of known duration. We show that if a system cannot execute arbitrary large sets of events without progressing time, unfolding up to some bounded depth is sufficient.

Definition 7 (time progress). *An STPN $\mathcal{N}$ guarantees* time progress *iff there exists $\delta \in \mathbb{Q}_{>0}$ such that $\forall t \in T, i \in \mathbb{N}$, and for every time word $u = \langle t_1, d_1\rangle \dots \langle t^i, \theta_1\rangle \dots \langle t^{i+1}, \theta_2\rangle \dots \langle t_k, d_k\rangle \in \mathcal{L}(\mathcal{N})$ where t^i denotes the i^{th} occurrence of t, we have $\theta_2 - \theta_1 \geq \delta$.*

Time progress is close to non-Zenoness property, and is easily met (e.g., if no transition has an earliest firing time of 0). Any execution of duration Δ of an STPN that guarantees time progress is a sequence of at most $|T| \cdot \lceil \frac{\Delta}{\delta} \rceil$ transitions.

As in processes, unfoldings will contain occurrences of transitions firings (a set of events E), and occurrences of places fillings (a set of conditions B). We associate to each event $e \in E$ positive real valued variables $\mathsf{doe}(e)$, $\mathsf{dof}(e)$ and $\theta(e)$ that respectively define the enabling, firability and effective firing date of the occurrence of transition $\mathsf{tr}(e)$ represented by event e. Similarly, we associate to each condition b positive real valued variables $\mathsf{dob}(b)$ and $\mathsf{dod}(b)$ that respectively represent the date of birth of the token in place $\mathsf{pl}(b)$, and the date at which the token in place $\mathsf{pl}(b)$ is consumed. We denote by $\mathsf{var}(E, B)$ the set of variables $\bigcup_{e\in E} \mathsf{doe}(e) \cup \mathsf{dof}(e) \cup \theta(e) \cup \bigcup_{b\in B} \mathsf{dob}(b) \cup \mathsf{dod}(b)$ (with values in

$\mathbb{R}_{\geq 0}$). A *constraint* over $\mathsf{var}(E, B)$ is a boolean combination of atoms of the form $x \bowtie y$, where $x \in \mathsf{var}(E, B)$, $\bowtie \in \{<, >, \leq, \geq\}$ and y is either a variable from $\mathsf{var}(E, B)$ or a constant value. A set of constraints C over a set of variables V is *satisfiable* iff there exists at least one valuation $v : V \to \mathbb{R}$ such that replacing each occurrence of each variable x by its valuation $v(x)$ yields a tautology. We denote by $\textsc{Sol}(C)$ the set of valuations that satisfy C.

Definition 8 (unfolding). *A* (structural) unfolding *of an STPN $\mathcal{N}$ is a pair $\mathcal{U} \triangleq \langle E, B\rangle$ where E is a set of events and B a set of conditions.*

Unfoldings can be seen as processes with branching. As for processes, each event $e \in E$ is a pair $e = \langle {}^\bullet e, \mathsf{tr}(e)\rangle$ where ${}^\bullet e \subseteq B$ is the set of predecessor conditions of e (the conditions needed for e to occur). A condition $b \in B$ is a pair $b \triangleq \langle {}^\bullet b, \mathsf{pl}(b)\rangle$ where ${}^\bullet b \subseteq E$ is the predecessor of b, i.e., the event that created condition b. We assume a dummy event $\perp$ that represents the origin of the initial conditions in an unfolding. Function ${}^\bullet()$, $()^\bullet$, $\mathsf{pl}()$ and $\mathsf{tr}()$ keep the same meaning as for time processes. The main change between processes and unfoldings is that conditions may have several successor events. Using relations $\prec$ and $\preceq$ as defined for processes, we define the *causal past* of $e \in E$ as $\uparrow e \triangleq \{e' \in E \mid e' \preceq e\}$. A set of events $E' \subseteq E$ is *causally closed* iff $\forall e \in E', \uparrow e \subseteq E'$. We extend this notion to conditions. Two events e, e' are in *conflict*, and write $e \sharp' e$, iff ${}^\bullet e \cap {}^\bullet e' \neq \varnothing$. A set of events $E' \subseteq E$ is *conflict free* if it does not contain conflicting pairs of events. Two events e, e' are *competing* iff $\mathsf{tr}(e)^\bullet \cap \mathsf{tr}(e')^\bullet \neq \varnothing$ (they fill a common place).

Definition 9 (pre-processes of an unfolding). *A* pre-process *of a finite unfolding $\mathcal{U} = \langle E, B\rangle$ is a pair $\langle E', B'\rangle$ such that $E' \subseteq E$ is a maximal (i.e., there is no larger pre-process containing E', B'), causally closed and conflict free set of events, and $B' = {}^\bullet E' \cup E'^\bullet$. $\mathcal{PE}(\mathcal{U})$ denotes the set of pre-processes of $\mathcal{U}$.*

We say that a condition $b \in B$ is *maximal* in $\mathcal{U} = \langle E, B\rangle$ or in a pre-process of $\mathcal{U}$ when it has no successor event ($b^\bullet = \varnothing$), and denote the set of maximal conditions of B by $\mathsf{max}(B)$. As for time processes construction, given a finite pre-process $\langle E', B'\rangle \in \mathcal{PE}(\mathcal{U})$, and a place p of the considered STPN, we denote by $\mathsf{last}(p, E', B')$ the maximal occurrences of place p w.r.t. $\prec$ in $\langle E', B'\rangle$. A *cut* of a pre-process is an unordered set of conditions. We denote by $Cuts(E, B)$ the set of cuts of pre-process $\langle E, B\rangle$.

Unfolding an STPN up to depth K is performed inductively, without considering time. We will then use this structure to find processes. Timing issues will be considered through addition of constraints on occurrence dates of events.

Structural unfolding: Following [12], we inductively build unfoldings $\mathcal{U}_0, \ldots, \mathcal{U}_K$. Each step k adds new events at depth k and their postset to the preceding unfolding $\mathcal{U}_{k-1}$. We start with the initial unfolding $\mathcal{U}_0 \triangleq \langle \varnothing, B_0\rangle$ where $B_0 = \{\langle \perp, p\rangle \mid p \in m_0\}$. Each induction step that builds $\mathcal{U}_{k+1}$ from $\mathcal{U}_k$ adds new events and conditions to $\mathcal{U}_k$ as follows. Letting $\mathcal{U}_k = \langle E_k, B_k\rangle$ be the unfolding obtained at step k, we have $\mathcal{U}_{k+1} = \langle E_k \cup \hat{E}, B_k \cup \hat{B}\rangle$ where $\hat{E} \triangleq \{\langle B, t\rangle \in (2^{B_k} \times T) \setminus E_k \mid \exists \langle X, Y\rangle \in \mathcal{PE}(\mathcal{U}_k), B \subseteq Cuts(X, Y), {}^\bullet t = \mathsf{pl}(B)\}$, and $\hat{B} \triangleq \{\langle e, p\rangle \in \hat{E} \times T \mid e = \langle B, t\rangle \in \hat{E} \wedge p \in t^\bullet\}$. Intuitively, $\hat{E}$ adds an occurrence of a transition if its preset is contained in the set of conditions representing

the last occurrences of places contained in some pre-process of $\mathcal{U}_k$, and $\hat{B}$ adds the conditions produced by $\hat{E}$.

The structural unfolding of an STPN does not consider timing issues nor blockings. Hence, an (untimed) pre-process of $\mathcal{PE}(\mathcal{U}_K)$ need not be the untimed version of a time process obtained from a word in $\mathcal{L}(\mathcal{N})$. Indeed, urgent transitions can forbid firing of other conflicting transitions. Similarly, blockings prevent an event from occurring as long as a condition in its postset is filled. They may even prevent events in a pre-process from being executed if a needed place is never freed. We will show later that, once constrained, time processes of $\mathcal{N}$ are only prefixes of pre-processes in $\mathcal{PE}(\mathcal{U}_K)$ with associated timing function. To introduce timing aspects, we now attach constraints on events and conditions of pre-processes as follows:

Constraints: Let $\mathcal{U}_K = \langle E_K, B_K \rangle$ be the unfolding of an STPN $\mathcal{N}$ up to depth K, and let $E \subseteq E_K$ be a conflict free and causally closed set of events, and $B = {}^\bullet E \cup E^\bullet$ (B is contained in B_K). We define $\Phi_{E,B}$ as the set of constraints attached to events and conditions in E, B, assuming that executions of $\mathcal{N}$ start at a fixed date d_0. Constraints must be set to guarantee that occurrence dates of events are compatible with the earliest and latest firing times of transitions in $\mathcal{N}$, and that urgency or blocking is never violated. Let us first define the constraints associated with each condition $b = \langle e, p \rangle$. Recalling that variable $\mathsf{dob}(b)$ represents the date at which condition b is created, $\Phi_{E,B}$ must impose that for every $b \in B_0$, $\mathsf{dob}(b) = d_0$.

For all other conditions $b = \langle e, p \rangle$, as the date of birth is exactly the occurrence date of e, we set $\mathsf{dob}(b) = \theta(e)$ for every $b = \langle e, p \rangle$. Despite this equality, we will use both variables $\theta(e)$ and $\mathsf{dob}(b)$ for readability reasons. Recall that $\mathsf{dod}(b)$ is a variable that designates the date at which a place is emptied by some transition firing, $\mathsf{dod}(b)$ is hence the occurrence date of an event that has b as predecessor. Within a conflict free set of events, this event is unique. In the considered subset of conditions B, several conditions may represent fillings of the same place, and B can hence be partitioned into $B_1 \uplus B_2 \uplus \cdots \uplus B_{|P|}$, where conditions in B_i represent fillings of place p_i. Due to blocking semantics, all conditions in a particular subset $B_i = \{b_{i,1}, b_{i,2}, \ldots, b_{i,k}\}$ must have disjoint existence dates, that is for every $j, j' \in \{1, 2, \ldots, k\}$ with $j \neq j'$, the intersection between $[\mathsf{dob}(b_{i,j}), \mathsf{dod}(b_{i,j})]$ and $[\mathsf{dob}(b_{i,j'}), \mathsf{dod}(b_{i,j'})]$ is either empty, or limited to a single value. This constraint can be encoded by the disjunction:

$$\mathsf{no\text{-}overlap}(b_{i,j}, b_{i,j'}) \triangleq \begin{cases} \mathsf{dod}(b_{i,j}) \leq \mathsf{dob}(b_{i,j'}) \vee \mathsf{dod}(b_{i,j'}) \leq \mathsf{dob}(b_{i,j}) & \text{if } {b_{i,j}}^\bullet \neq \varnothing \wedge {b_{i,j'}}^\bullet \neq \varnothing, \\ \mathsf{dod}(b_{i,j}) \leq \mathsf{dob}(b_{i,j'}) & \text{if } {b_{i,j}}^\bullet \neq \varnothing \wedge {b_{i,j'}}^\bullet = \varnothing, \\ \mathsf{dod}(b_{i,j'}) \leq \mathsf{dob}(b_{i,j}) & \text{otherwise.} \end{cases}$$

Note that if $b_j \preceq b_{j'}$, then the constraint among events and transitions immediately ensures $\mathsf{dob}(b_{j,i}) \leq \mathsf{dod}(b_{j,i}) \leq \mathsf{dob}(b_{j',i}) \leq \mathsf{dod}(b_{j',i})$. However, we need to add a consistency constraint for every pair of concurrent conditions $b_{i,j}, b_{i,j'}$ that belong to the same B_i. Hence, calling $I(b_{i,j}, E, B)$ the set of conditions that represent the same place as $b_{i,j}$ and are concurrent with $b_{i,j}$ in $\langle E, B \rangle$, we have to ensure the constraint $\mathsf{non\text{-}blocking}(b_{i,j}) \triangleq \bigwedge_{b_{i,j'} \in I(b_{i,j}, E, B)} \mathsf{no\text{-}overlap}(b_{i,j}, b_{i,j'})$.

In words, condition $b_{i,j}$ does not hold during the validity dates of any concurrent condition representing the same place. In particular, a time process of $\mathcal{N}$ cannot contain two maximal conditions with the same place.

Let us now consider the constraints attached to events. An event $e = \langle B, t\rangle$ is an occurrence of a firing of transition t that needs conditions in B to be fulfilled to become enabled. Calling $\mathsf{doe}(e)$ the date of enabling of e, we necessarily have $\mathsf{doe}(e) = \max\{\mathsf{dob}(b) \mid b \in B\}$. Event e is firable at least $\mathsf{eft}(t)$ time units, and at most $\mathsf{lft}(t)$ time units after being enabled. We hence have $\mathsf{doe}(e) + \mathsf{eft}(t) \leq \mathsf{dof}(e) \leq \mathsf{doe}(e) + \mathsf{lft}(t)$. However, execution of e does not always occur immediately when e is firable. Execution of e occurs after e is firable, as soon as the places filled by e are empty, i.e., e occurs at a date $\theta(e)$ that guarantees that no place in $t^\bullet$ is occupied. This is guaranteed by attaching to every event e the constraints $\theta(e) = \mathsf{dob}(b_1), \theta(e) = \mathsf{dob}(b_2), \ldots, \theta(e) = \mathsf{dob}(b_k)$, where $\{b_1, b_2, \ldots b_k\} = e^\bullet$, and constraints $\mathsf{non\text{-}blocking}(b_1), \mathsf{non\text{-}blocking}(b_2), \ldots, \mathsf{non\text{-}blocking}(b_k)$. Last, as semantics of STPNs is urgent, once firable, e has to fire at the earliest possible date. This is encoded by the constraint $\theta(e) = \min\{x \in \mathbb{R}_{\geq 0} \mid x \notin]\mathsf{dob}(b), \mathsf{dod}(b)[$ for some $b \in \bigcup I(b_i) \wedge x \geq \mathsf{dof}(e)\}$. Figure 2 shows the effect of blocking and possible free firing dates for some event with a condition b in its postset. Horizontal lines represent real lines, and intervals values in interval $[\mathsf{dob}(b_i), \mathsf{dod}(b_i)]$ for $i \in 0, 1, 2$. Suppose that $I(b) = \{b_0, b_1, b_2\}$. Then $[\mathsf{dob}(b), \mathsf{dod}(b)]$ have to be fully inscribed in one of these thick segments. An event with b in its postset can occur only at dates contained in these thick segments.

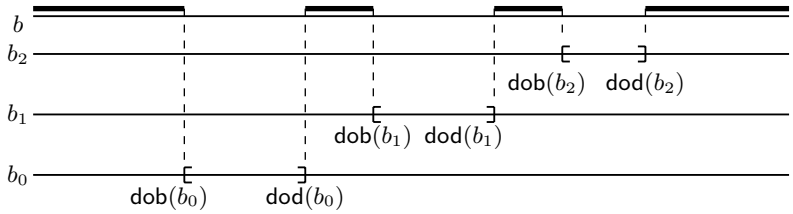

Fig. 2: Constraints on dates of birth of tokens in a shared place.

Written differently,

$$\theta(e) = \begin{cases} \mathsf{dof}(e) \text{ if } \bigwedge_{b \in I(b_1) \cup \ldots I(b_k)} \mathsf{dof}(e) \leq \mathsf{dob}(b), \text{ and} \\ \min\{\mathsf{dod}(b) \mid \forall b' \in \bigcup_{b_i \in e^\bullet} I(b_i), \mathsf{dod}(b) \notin]\mathsf{dob}(b'), \mathsf{dod}(b')[\} \text{ otherwise.} \end{cases}$$

This formula can be translated in boolean combinations of inequalities over variables of $\mathsf{var}(E, B)$. Similarly, event $e = \langle B, t\rangle$ must occur before all its conflicting events. If an event e' in conflict with e is executed, at least one condition in B is consumed, and e cannot occur in a time process containing e'. We hence need the additional constraint $\bigwedge_{e' \sharp e} \mathsf{notMoreUrg}(e, e')$ to guarantee that there exists no other event that is forced to occur before e due to urgency. We define $\mathsf{notMoreUrg}(e, e')$ as the following constraint:

$\mathsf{notMoreUrg}(e, e') \triangleq \theta(e) \geq \mathsf{doe}(e') + \mathsf{lft}(\mathsf{tr}(e')) \Rightarrow \mathsf{tiled}(e, e') \vee \bigvee_{e'' || e} \mathsf{preempts}(e', e'')$

where $\mathsf{tiled}(e,e') \triangleq \mathsf{free}(e') \cap [\mathsf{doe}(e') + \mathsf{lft}(\mathsf{tr}(e')), \theta(e)] = \varnothing$, $e''||e$ refers to events that are concurrent with e in the considered set of events E, $\mathsf{free}(e') = \mathbb{R}_{\geq 0} \setminus \{[\mathsf{dob}(b), \mathsf{dod}(b)] \mid \exists b' \in e'^{\bullet}, b \in I(b')\}$ is the set of intervals in which places attached to conditions in $e'^{\bullet}$ are empty, and $\mathsf{preempts}(e',e'') \triangleq \theta(e'') \leq \min(]\mathsf{doe}(e') + \mathsf{lft}(\mathsf{tr}(e')), \theta(e)[\cap \mathsf{free}(e'))$ means e'' disabled e' by consuming a condition in ${}^{\bullet}e''$.

Constraint $\mathsf{notMoreUrg}(e,e')$ means that if e' is in conflict with e, then at least one condition in ${}^{\bullet}e'$ is consumed before e' can fire, or if e' becomes firable before e fires, the urgent firing of e' is delayed by blockings so that e can occur. As for constraint attached to blockings, $\mathsf{notMoreUrg}(e,e')$ can be expressed as a boolean combination of inequalities. One can also notice that $\mathsf{notMoreUrg}(e,e')$ can be expressed without referring to variables attached to event e' nor $e'^{\bullet}$, as $\mathsf{doe}(e') = \max_{b_i \in {}^{\bullet}e'} \mathsf{dob}(b_i)$ and the intersection of $I(b)$ and $e'^{\bullet}$ is void.

For causally closed sets of events and conditions $E \cup B$ contained in some pre-process of $\mathcal{U}_K$, the constraint $\Phi_{E,B}$ applying on events and conditions of $E \cup B$ is now defined as $\Phi_{E,B} = \bigwedge_{x \in E \cup B} \Phi_{E,B}(x)$ where:

$$\forall b \in B, \Phi_{E,B}(b) = \mathsf{non\text{-}blocking}(b) \wedge \begin{cases} \mathsf{dob}(b) = d_0 \text{ if } b \in B_0, \text{ and } b \text{ is maximal,} \\ \mathsf{dob}(b) = d_0 \wedge \mathsf{dob}(b) \leq \mathsf{dod}(b) \text{ if } b \in B_0, \\ \mathsf{dob}(b) = \theta({}^{\bullet}b) \text{ if } b \notin B_0 \text{ and } b \text{ is maximal,} \\ \mathsf{dob}(b) = \theta({}^{\bullet}b) \wedge \mathsf{dob}(b) \leq \mathsf{dod}(b) \text{ otherwise.} \end{cases}$$

$$\forall e \in E, \Phi_{E,B}(e) = \begin{cases} & \mathsf{doe}(e) = \max_{b \in {}^{\bullet}e} \mathsf{dob}(b) \\ \wedge & \mathsf{doe}(e) + \mathsf{eft}(\mathsf{tr}(e)) \leq \mathsf{dof}(e) \leq \mathsf{doe}(e) + \mathsf{lft}(\mathsf{tr}(e)) \\ \wedge & \mathsf{dof}(e) \leq \theta(e) \wedge \bigwedge_{b \in {}^{\bullet}e} \mathsf{dod}(b) = \theta(e) \\ \wedge & \bigwedge_{b \in e^{\bullet}} \theta(e) = \mathsf{dob}(b) \\ \wedge & \bigwedge_{e' \sharp e} \mathsf{notMoreUrg}(e,e') \end{cases}$$

We can now define symbolic processes, and show how instantiation of their variables define time processes of $\mathcal{N}$. Roughly speaking, a symbolic process is a prefix of a pre-process of $\mathcal{U}_K$ (it is hence a causal net) decorated with a *satisfiable* set of constraints on occurrence dates of events. Before formalizing symbolic processes, let us highlight three important remarks. **Remark 1:** an unfolding up to depth K misses some constraints on occurrence dates of events due to blockings by conditions that do not belong to $\mathcal{U}_K$ but would appear in some larger unfolding $\mathcal{U}_{K'}$, with $K' > K$. We will however show (Prop. 1 and 2 that with time progress assumption, unfolding $\mathcal{N}$ up to a sufficient depth guarantees that all constraints regarding events with $\theta(e) \leq D$ are considered. This allows to define symbolic processes representing the time processes of $\mathcal{N}$ that are executable in less than D time units. **Remark 2:** unfoldings consider depth of events, and not their dates. Hence, if a process contains an event e occurring at some date greater than d, and another event e' that belongs to the same pre-process and becomes urgent before date d, then e' must belong to the process, even if it lays at a greater depth than e. **Remark 3:** Every pre-process $\langle E, B \rangle$ of $\mathcal{U}_K$ equipped with constraint $\Phi_{E,B}$ is not necessarily a symbolic process. Indeed, some events in a pre-process might be competing for the same resource. Consider for instance the STPN of Figure 3-a). Its unfolding is represented in b), and two of its (symbolic)

processes in c) and d). For readability, we have omitted constraints. One can however notice that there exists no symbolic process containing two occurrences of transition t_3, because conditions p_4^1 and p_4^2 are maximal and represent the same place p_4.

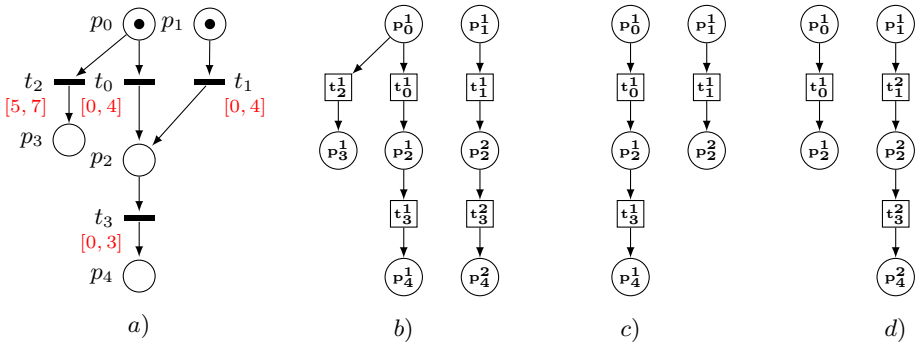

Fig. 3: An STPN with conflicts and blockings a), its symbolic unfolding b), and two of its symbolic processes c) and d).

Definition 10 (prefixes of an unfolding). *Let $SPP = \langle E, B\rangle$ be a pre-process of $\mathcal{U}_K$. A* symbolic prefix *of SPP is a triple $\langle E', B', \Phi_{E',B'}\rangle$ where $E' \subseteq E$ is a causally closed set of elements contained in E, and $B' = {}^\bullet E' \cup E'^\bullet$.*

Symbolic prefixes are causally closed parts of pre-processes, but their constraints inherited from the unfolding $\mathcal{U}_K$ may not be satisfiable.

Definition 11 (symbolic processes). *A* symbolic process *of $\mathcal{U}_K$ is a triple $\mathcal{E}^s = \langle E', B', \Phi_{E',B'}\rangle$ where $\langle E', B', \Phi_{E',B'}\rangle$ is a symbolic prefix of some pre-process $PP = \langle E, B\rangle$ of $\mathcal{U}_K$, $\Phi_{E',B'}$ is satisfiable, and E' is* maximal *w.r.t. urgent events firing in PP, that is for every $f \in B'^\bullet \cap E$, and letting $C_f = \mathsf{pl}^{-1}(f^\bullet) \cap B'$ denote the set of conditions whose place appears in the postset of e, the following constraint is not satisfiable.*

$$\Phi_{\max}(f) \triangleq \begin{cases} \Phi_{E',B'} \\ \wedge\ \theta(f) \leq \max_{e' \in E'} \theta(e') \ \text{(f fires before the last event in E')} \\ \wedge\ \mathsf{eft}(f) + \max_{b \in {}^\bullet f} \mathsf{dob}(b) \leq \theta(f) \ \text{(f is urgent)} \\ \wedge\ \bigvee_{X \in 2^{C_f}} \max_{x \in X} \mathsf{dod}(x) \leq \theta(f) \leq \min_{x \in C_f \setminus X} \mathsf{dob}(x) \\ \quad \text{(f is not blocked for the whole duration of the process)} \end{cases}$$

Intuitively, $\Phi_{\max}(f)$ means that f, that is not in the symbolic process, becomes urgent, is not blocked by conditions in B', and has to fire before the execution of the last event in E'. If $\Phi_{\max}(f)$ is satisfiable, then f should appear in the process. A crux in the construction of symbolic processes of $\mathcal{U}_K$ is to find appropriate maximal and causally closed sets of events with satisfiable constraints. This can be costly: as illustrated by the example of Figure 3, satisfiability of constraints is not monotonous: the constraints for processes in Fig 3$-c$) and d) are satisfiable. However, adding one occurrence of transition t_3 yields unsatisfiable constraints.

Satisfiability of a prefix of size n hence does not imply satisfiability of a larger prefix of size $n+1$. The converse implication is also false: if a constraint of a prefix of size n is not satisfiable, appending a new event may introduce, blockings, delay urgent transitions, yielding satisfiability of a constraint on a prefix of size $n+1$. So, satisfiability of constraints is not a criterion to stop unfolding.

Definition 12 (executions of symbolic processes). *Let $\mathcal{E}^s = \langle E, B, \Phi \rangle$ be a symbolic process of an unfolding $\mathcal{U}_K$. An* execution *of $\mathcal{E}^s$ is a time process $TP = \langle E, B, \theta \rangle$ where θ is a solution for Φ. For a chosen θ, we denote by $\mathcal{E}^s_\theta = \langle E, B, \theta \rangle$ the time process obtained from $\mathcal{E}^s$. $TP = \langle E, B, \theta \rangle$ is a* time process *of $\mathcal{U}_K$ if there exists a symbolic process $\mathcal{E}^s = \langle E, B, \Phi \rangle$ of $\mathcal{U}_K$ s.t. TP is an execution of $\mathcal{E}^s$.*

Informally, symbolic pre-processes select maximal conflict-free sets of events in an unfolding. Symbolic processes extract executable prefixes from symbolic pre-processes, and executions attach dates to events of symbolic processes to obtain time processes. In the rest of the paper, we respectively denote by $\mathcal{E}^s(\mathcal{U}_K)$ and by $\mathcal{E}(\mathcal{U}_K)$ the set of symbolic processes and time processes of $\mathcal{U}_K$.

We can now show that upon time progress hypothesis, unfoldings and their symbolic processes capture the semantics of STPNs with blockings. Given an STPN that guarantees time progress with a minimal elapsing of δ time units between successive occurrences of every transition, and given a maximal date D, we want to build an unfolding $\mathcal{U}^D$ of $\mathcal{N}$ that contains all events that might be executed before D, but also all places and events which may impact firing dates of these events. We can show that $\mathcal{U}^D$ is finite and its processes are of depth $H = \lceil \frac{D-d_0}{\delta} \rceil \cdot |T|$ at most.

Let $b = \langle e, p \rangle$ be a condition of an unfolding $\mathcal{U}_n$ obtained at step n. Let $\mathsf{block}(b)$ be the set of conditions that may occur in the same process as b, represent the same place, and are not predecessors or successors of b in any unfolding $\mathcal{U}_{n+k}$ obtained from $\mathcal{U}_n$. Clearly, dates of birth and death of conditions in $\mathsf{block}(b)$ may influence the date of birth and death of b, or even prevent b from appearing in the same process as some conditions in $\mathsf{block}(b)$. However, in general, $\mathsf{block}(b)$ need not be finite, and at step n, $\mathsf{block}(b)$ is not fully contained in a pre-process of $\mathcal{U}_n$. Fortunately, upon time progress assumption, we can show that elements of $\mathsf{block}(b)$ that can influence $\mathsf{dob}(b)$ appear in some bounded unfolding $\mathcal{U}_K$.

Proposition 1. *Let $\mathcal{N}$ be a STPN guaranteeing time progress of δ time units (between consecutive occurrences of each transition). For every date $D \in \mathbb{R}_{\geq 0}$ and condition b in an unfolding $\mathcal{U}_n$, there exists $K \geq n$ s.t. $\{b' \in \mathsf{block}(b) \mid \mathsf{dob}(b') \leq D\}$ is contained in $\mathcal{U}_K$.*

This proposition means that if some event cannot occur at $\mathsf{dof}(e)$ due to a blocking, then one can discover all conditions that prevent this firing from occuring in a bounded extension of the current unfolding.

Proposition 2. *Let $\mathcal{N}$ be a STPN guaranteeing time progress of δ time units. The set of time processes executable by $\mathcal{N}$ in D time units are prefixes of time processes of $\mathcal{U}_K$, with $K = \lceil \frac{D}{\delta} \rceil \cdot |T|^2$ containing only events with date $\leq D$.*

4 Realizability of schedules

We can now address realizability of a schedule S, i.e., a high-level description of operations of a system and of their timing constraints can be realized by a system represented as a STPN $\mathcal{N}$ depicting low-level operations and distributions over possible delays between enabledness and firing of transitions. The connection between operations in S and $\mathcal{N}$ is defined via a realization function.

Definition 13 (realization function). *A* realization function *for a schedule S and an STPN $\mathcal{N}$ is a map $r : \mathcal{A} \to 2^T$ that associates a subset of transitions from T to each letter of $\mathcal{A}$, and such that $\forall a \neq a' \in \mathcal{A}, r(a) \cap r(a') = \varnothing$.*

A realization function describes which low-level actions implement a high-level operation of a schedule. Each letter a from $\mathcal{A}$ can be interpreted as an operation performed through the firing of any transition from the subset of transitions $r(a)$. Allowing $r(a)$ to be a subset of T provides some flexibility in the definition of schedules: in a production cell, for example, a manufacturing step a for an item can be implemented by different processes on different machines. Similarly, in a train network, a departure of a train from a particular station in the schedule can correspond to several departures using different tracks, which is encoded with several transitions in an STPN. Realization functions hence relate actions in schedules to several transitions in an STPN. The condition on realization functions prevents ambiguity by enforcing each transition to appear at most once in the image of r. Note that $r(\mathcal{A}) \subseteq T$, that is the realization of a schedule may need many intermediate steps that are depicted in the low-level description of a system, but are not considered in the high-level view provided by a schedule. We will call transitions that belong to $r(\mathcal{A})$ *realizations* of $\mathcal{A}$.

Definition 14 (embedding, realizability). *Let $S = \langle N, \to, \lambda, C\rangle$ be a schedule, $\mathcal{E}^s = \langle E, B, \Phi\rangle$ be a symbolic process of $\mathcal{N}$ and $r : N \to T$ be a realization function. We say that S* embeds *into $\mathcal{E}^s$ (w.r.t. r and d) and write $S \hookrightarrow \mathcal{E}^s$ iff there exists an injective function $\psi : N \to E$ such that:*

$$\begin{cases} \forall n \in N, \mathsf{tr}\big(\psi(n)\big) \in r\big(\lambda(n)\big) \\ \forall \langle n, n'\rangle \in \to, \psi(n) \preceq \psi(n') \\ \nexists f \leq \psi(\min(n)), \mathsf{tr}(f) \in r(\mathcal{A}) \\ \forall e \leq f \leq g, e = \psi(n) \wedge g = \psi(n'') \wedge \mathsf{tr}(f) \in r(\mathcal{A}) \\ \qquad\qquad \Rightarrow \exists n', f = \psi(n') \wedge n \to^* n' \to^* n'' \end{cases}$$

S embeds in $\mathcal{E}^s$ iff there is a way to label every node n of S by a letter from $r\big(\lambda(n)\big)$ and obtain a structure that is contained in some restriction of a prefix of $\mathcal{E}^s$ to events that are realizations of actions from $\mathcal{A}$ and to a subset of its causal ordering. Note that there can be several ways to embed S into a process of $\mathcal{N}$.

Definition 15 ((boolean) realizability). *Let d be a dating function for a schedule S, r be a realization function. S is* realizable *by $\mathcal{E}^s$ (w.r.t. r and d) iff there exists an embedding ψ from S to $\mathcal{E}^s$, and furthermore, $\Phi_{\psi,S,d} \triangleq \Phi \wedge \bigwedge_{n\in N} \theta\big(\psi(n)\big) = d(n)$ is satisfiable. S is* realizable *by $\mathcal{N}$ (w.r.t. r and d) iff there exists a symbolic process $\mathcal{E}^s$ such that S is* realizable *by $\mathcal{E}^s$.*

We write $\mathcal{E}^s \models S$ when S is realizable by $\mathcal{E}^s$, and $\mathcal{N} \models S$ when S is realizable by $\mathcal{N}$. An algorithm to compute a set $\Psi_{S,\mathcal{E}^s}$ of embeddings of a schedule S in a process $\mathcal{E}^s$ is provided in the extended version. Once $\Psi_{S,\mathcal{E}^s}$ is obtained, it remains to show that for at least one embedding $\psi \in \Psi_{S,\mathcal{E}^s}$, $\Phi_{\psi,S,d}$ is satisfiable to prove that S is realizable by $\mathcal{E}^s$. We can then compute the set of symbolic processes $\mathcal{E}^S \triangleq \{\mathcal{E}^s_0, \mathcal{E}^s_1, \ldots, \mathcal{E}^s_{N-1}\}$ of $\mathcal{U}_K$ that embed S and similarly for each $\mathcal{E}^s_i \in \mathcal{E}^S$ the set of possible embedding functions $\Psi_i \triangleq \{\psi_{i,0}, \psi_{i,1}, \ldots, \psi_{i,N_i-1}\}$ for which the constraints $\Phi_{\psi_{i,j},S,d}$ are satisfiable.

To illustrate the construction of unfoldings and of processes, let us consider the example of figure 4. This toy example depicts two train carousels that serve stations. Line 1 serves stations A, B and C, and line 2 serves stations D, B' and C'. Both lines share a common track portion between stations B, C and B', C', and line 1 uses two trains. A possible required schedule (top left of the figure) is that one train leaves every 10 time units from station A on line 1, starting from date 10, and one train leaves station C' every 10 time units, but starting from date 15. Departures from A are nodes labeled by d_A and departures from C' are nodes labeled by $d_{C'}$. The bottom left picture shows the aspect of both lines and stations. The center picture is an STPN model for this example, and we set $r(d_A) = \{t_5\}$ and $r(d_{C'}) = \{t_9\}$. We do not precise distributions, and focus on the structural unfolding, on the right of the figure. Note that the topmost occurrence of place OK, that plays the role of a boolean flag in a critical section can be both consumed by occurrences t_1^1 and t_1^2 of transition t_1, which is a standard conflict. However, as events t_4^1 and t_4^2 both output a token in place A, their firing times may influence one another even though they are not in conflict.

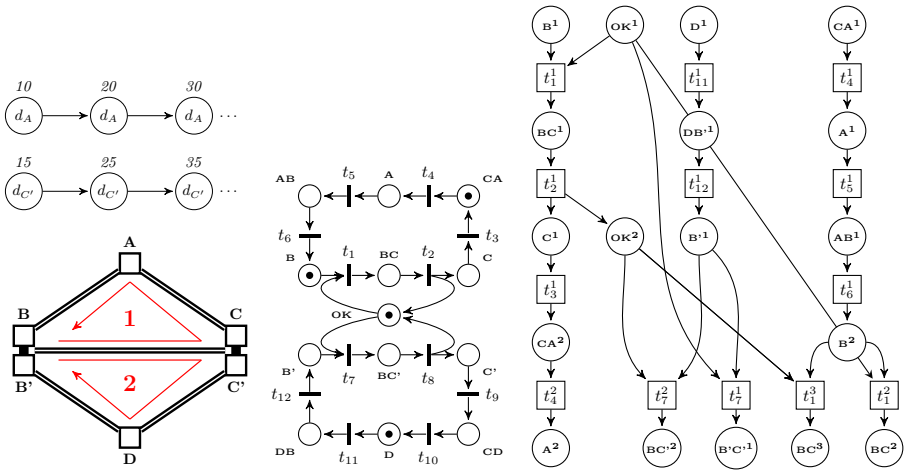

Fig. 4: A toy example: realizability of a partial schedule for two train carrousels with shared track portions.

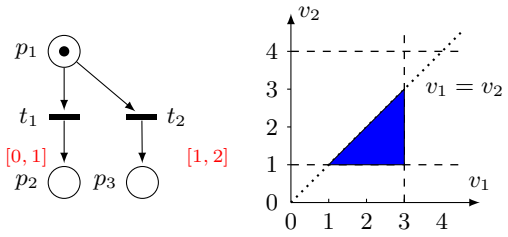

Fig. 5: *a*) An example STPN *b*) A domain for $\tau(t_1), \tau(t_2)$ allowing firing of t_2.

Let us now show that boolean realizability is not always a precise enough notion to characterize feasability of a schedule. Consider the STPN of Figure 5, and the two symbolic processes: one in which transition t_1 fires, and another one in which t_2 fires. The probability of the first process is the probability that a value v_1 sampled to assign a TTF for t_1 is smaller or equal to another value v_2 sampled independently to assign a TTF for t_2. Clearly, the probability that $v_1 \leq v_2$ is equal to the probabilty that $v_1 \in [0, 1]$. The probability of the second process is equal to the probability that $v_1 \geq v_2$, but the set of values allowing this inequality is restricted to a single point $v_1 = 1, v_2 = 1$. Conforming to continous probability distribution semantics, the probability of this point is equal to zero. A schedule composed of a single node n with date 1 such that $r\big(\lambda(n)\big) = \{t_2\}$ is realizable according to Definition 14, but with *null probability*. A more accurate notion of realizability is to require that schedules embed into symbolic processes of $\mathcal{U}_K$ with strictly positive probability.

This raises a second issue: requiring a schedule to be realized with an exact timing also leads to realizations with null probabilities. Consider the former example: a schedule composed of a single node n, a realization function r s.t. $r(\lambda(n)) = \{t_2\}$, and a dating function d s.t. $d(n) = 2$. Assign interval $[0, 3]$ to transition t_1 in the STPN of Figure 5-a) and interval $[1, 4]$ to transition t_2. The probability that t_2 fires from the initial marking is equal to the probability that $v_1 \geq v_2$, which is not null (we will explain later how to compute the probability of such domain and the joint probability of v_1, v_2), and is equal to the probability of the domain for values of v_1, v_2 depicted by the colored zone in Figure 5-b). However, within this continuous domain of possible values, the probability to fire t_2 exactly at precise date 2 as required by dating function d is still null. We hence consider realizability of a schedule up to some admissible imprecision α. Once an injection ψ from a schedule S to a symbolic process $\mathcal{E}^s$ is found, the constraint to meet becomes: $\Phi_{\psi,S,d\pm\alpha} = \Phi \wedge \bigwedge_{n\in N} \max(d(n) - \alpha, 0) \leq \theta(\psi(n)) \leq d(n) + \alpha$.

Definition 16 (probabilistic realizability). *A schedule with maximal date D is* realizable with non-null probability *iff there exists an embedding ψ of S into a symbolic process $\mathcal{E}^s$ of $\mathcal{U}_K$ s.t. $\mathbb{P}(\mathcal{E}^s \wedge \textsc{Sol}(\Phi_{\psi,S,d\pm\alpha})) > 0$.*

Intuitively, this definition requires that a symbolic process embeds S, and that the probability that this process is executed and satisfies all timing constraints imposed by the STPN and by the dating function is non-null. This

probability can be evaluated using a transient execution tree, as proposed in [13]. Roughly speaking, nodes of this tree are abstract representations of time domains for sampled values attached to enabled transitions (this is the usual notion of state class, already used in [3, 15] to analyze time Petri nets). In addition to state classes, transient tree nodes contain abstract representations of probability distributions. If the definition of distribution is appropriately chosen, for instance, using truncated sums of exponentials of the form

$$f(x) \;=\; \begin{cases} \sum c_k x^{a_k} e^{-\lambda_k x} \text{ if } x \in [a,b] \\ 0 \text{ otherwise} \end{cases}$$

then the distributions obtained by projection, multiplication, or variable elimination can still be encoded as sums of exponentials, and memorized using a finite set of parameters. The probability to fire a particular transition from a state and move to a successor node is computed as an integration over the time domains allowing this transition to fire first. The children of a node give a probabilistic distribution on possible classes of successor states. As time progress is guaranteeed in our model, a finite tree representing executions of an STPN or of one of its processes up to some bounded duration can be built. As explained in [13], the sum of probabilities attached to all paths of the tree can be used to compute the probability of some properties. In our case, the sum of probabilities of all paths that end with the execution of a chosen symbolic process gives the probability to realize this process. Details on construction of a transient tree are provided in the extended version.

5 Conclusion

Related work: we have addressed realizability of partially ordered timed schedules by timed and stochastic concurrent systems with blocking semantics. Realizability in a timed setting has been addressed as a timed game problem [11], with a boolean answer. The objective in this work is to check whether a player in a timed game has a strategy to ensure satisfaction of a formula written in a timed logic called Metric Interval Temporal Logic. Brought back to the setting of realizability of schedules, the work of [11] can be used to answer a boolean realization question, by translating a schedule to a formula. However, the work of [11] lies in an interleaved setting: a sequential formula cannot differentiate interleaved and concurrent actions. It does not address randomness in systems and hence cannot quantify realizability. Scheduling of train networks was already addressed as a contraint satisfaction problems [8]. The input of the problem is given as an alternative graph (that can be seen as some kind of unfolding of a systems's behavior, decorated with time constraints. The algorithms in [8] use a branch and bound algorithm to return an optimal schedule for the next 2 hours of operation of a train network, but do not consider randomness.

Realizability is also close to diagnosis. Given a log (a partial observation of a run of a system), and a model for this system, diagnosis aims at finding all possible runs of the model of the system whose partial observation complies with the

log. Considering a log as a schedule, the ability to compute a diagnosis implies realizability of this high-level log by the model. Diagnosis was addressed for stochastic Petri nets in [1]. In this work, the likelihood of a process that complies with an observation is evaluated, and time is seen as a sequence of discrete instants. Diagnosis was addressed for timed Petri nets in [4], where unfolding of a timed Petri net is built to explain an observed log. [10] proposes temporal patterns called chronicles that represent possible evolutions of an observed system. A chronicle is a set of events, linked together by time constraints. The diagnosis framework explains stream of time-stamped events as combinations of chronicles. Assembling chronicles is some kind of timed unfolding. However, event streams are not a concurrent model, and chronicles extraction does not consider randomness.

Schedulability can also be seen as conformance of an expected behavior (the schedule) to an implementation (the Petri net model). Conformance was defined as a timed input output conformance relation (tIOCO) relation between timed input/output automata in [14]. More precisely, $\mathcal{A}_1$ tIOCO $\mathcal{A}_2$ iff after some timed word, the set of outputs produced by $\mathcal{A}_1$ is included in the outputs produced by $\mathcal{A}_2$. This relation cannot be verified in general (as inclusion of timed automata languages is not decidable), but can be tested. Boolean realizability can be seen as some kind of conformance test. Note however that tIOCO is defined for an interleaved timed model without probabilization of transitions.

Assessment: The techniques described in this work first build an unfolding $\mathcal{U}_K$ up to a depth K that depends on the maximal date appearing in the schedule, find symbolic processes of $\mathcal{U}_K$ that embed the schedule, and then check that at least one of them has non-null probability. So far, we did not consider complexity issues. The size of an unfolding can grow exponentially w.r.t. its depth. Checking satisfiability of a set of constraints with disjunctions can also be costly. Satisfiability of constraints is not monotonous and hence cannot be used to stop unfolding. However, embedding verification and unfolding can be done jointly: one can stop a branch of unfolding as soon as a schedule does not embed in the pre-process on this branch. Most of the constraints presented in this paper can be simplified, and refer mainly to event variables. One can also notice that atoms in constraints are rather simple inequalities, which could simplify their verification. Computation of realization probability for processes can also be improved. We use the transient tree construction of [13], that builds a symbolic but *interleaved* representation of some processes. This is obviously very costly. We are currently investigating ways to evaluate probabilities of symbolic processes in a non-interleaved setting.

As future work, we would like to implement and improve this realizability verification framework, and use it as a basis to prove more properties. For instance, it might be interesting to prove that a schedule can be realized while ensuring that the overall sum of delays w.r.t the expected schedule does not exceed some threshold. Another improvement would be to provide means to compute an exact value for the realization probability. We are also interested in the design of controllers that maximize the probability of realization.

References

1. A. Aghasaryan, E. Fabre, A. Benveniste, R. Boubour, and C. Jard. Fault detection and diagnosis in distributed systems: An approach by partially stochastic Petri nets. *Discrete Event Dynamic Systems*, 8(2):203–231, 1998.
2. T. Aura and J. Lilius. Time processes for time Petri nets. In *ICATPN'97*, volume 1248 of *LNCS*, pages 136–155, 1997.
3. B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE Trans. Software Eng.*, 17(3):259–273, 1991.
4. T. Chatain and C. Jard. Symbolic diagnosis of partially observable concurrent systems. In *FORTE'04*, volume 3235 of *LNCS*, pages 326–342, 2004.
5. T. Chatain and C. Jard. Time supervision of concurrent systems using symbolic unfoldings of time Petri nets. In *FORMATS'05*, volume 3829 of *LNCS*, pages 196–210. Springer, 2005.
6. T. Chatain and C. Jard. Complete finite prefixes of symbolic unfoldings of safe time Petri nets. In *ICATPN'06*, pages 125–145. Springer, 2006.
7. J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev. Synthesizing Petri nets from state-based models. In *ICCAD'95*, pages 164–171, 1995.
8. A. D'Ariano, D. Pacciarelli, and M. Pranzo. A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operational Research*, 183(2):643–657, 2007.
9. A. D'Ariano, M. Pranzo, and I.A. Hansen. Conflict resolution and train speed coordination for solving real-time timetable perturbations. *IEEE Trans. on Intelligent Transportation Systems*, 8(2):208–222, 2007.
10. C. Dousson. Extending and unifying chronicle representation with event counters. In *ECAI'02*, pages 257–261, 2002.
11. L. Doyen, G. Geeraerts, J.F. Raskin, and J. Reichert. Realizability of real-time logics. In *FORMATS'09*, volume 5813 of *LNCS*, pages 133–148, 2009.
12. J. Esparza, S. Römer, and W. Vogler. An improvement of McMillan's unfolding algorithm. *Formal Methods in System Design*, 20(3):285–310, 2002.
13. A. Horváth, M. Paolieri, L. Ridi, and E. Vicario. Transient analysis of non-Markovian models using stochastic state classes. *Performance Evaluation*, 69(7):315–335, 2012.
14. M. Krichen and S. Tripakis. Conformance testing for real-time systems. *Formal Methods in System Design*, 34(3):238–304, 2009.
15. D. Lime and O.H. Roux. Model checking of time Petri nets using the state class timed automaton. *Journal of Discrete Event Dynamic Systems*, 16(2):179–205, 2006.
16. K. L. McMillan. A technique of state space search based on unfolding. *Formal Methods in System Design*, 6(1):45–65, 1995.
17. R. Y. Rubinstein and D.P. Kroese. *Simulation and the Monte Carlo Method*. Wiley, 2 edition, 2008.
18. A.L. Semenov and A. Yakovlev. Verification of asynchronous circuits using time Petri net unfolding. In *DAC*, pages 59–62, 1996.